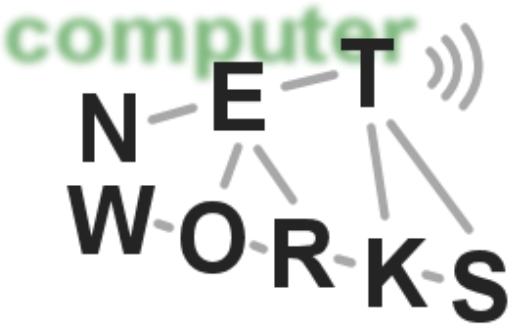
Cloud Computing – Part 1 of 3

Advanced Computer Networks

Summer Semester 2014



computer NETWORKS

Cloud Computing Overview

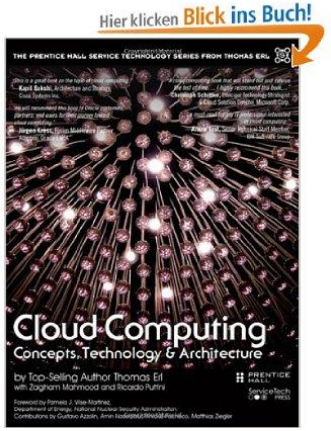
- Today's lecture:
  - Introduction
  - Basic principles and characteristics
- Today and next week:
  - Main standards
    - E.g., SDN, MapReduce, Hadoop File System, Hive, …
- Next week and 22/5:
  - Advances in research

Cloud Computing Overview

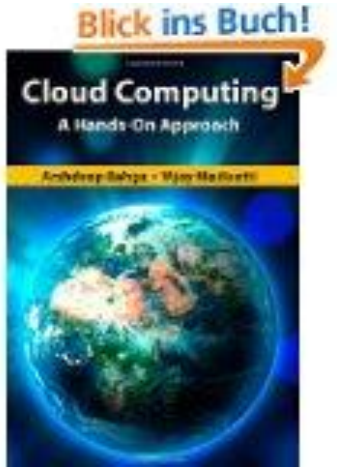
- 22/5 lecture:
  - Shortened to ~45 minutes
  - Afterwards: A guest talk by Ruediger Geib from Deutsche Telekom

Book Recommendations

- T. Erl, “Cloud Computing: Concepts, Technology & Architecture”, Prentice Hall
  - For a high-level view on Cloud Computing



- A. Bahga, “Cloud Computing: A Hands-On Approach”



CleanSky EU ITN Project

- An EU training project for young researchers starting in September 2014
- Project coordination done by our group: http://www.cleansky-itn.eu

- There will be several open PhD/PostDoc positions available throughout universities/enterprises in Europe
  - Göttingen, Trondheim (Norway), Helsinki (Finland), Heidelberg (Germany)

- Requirements for PhD positions:
  - Excellent Master's degree
  - You can not have lived for more than one year in the country you apply for over the past three years

- Research focus on green cloud computing strategies

Introduction to Cloud Computing

Today‘s lecture will cover…

- **What?**
  - What is Cloud Computing?
- **Why?**
  - Why do we need Cloud Computing?
- **How?**
  - How is the Cloud working?
    - (in parts next week(s))

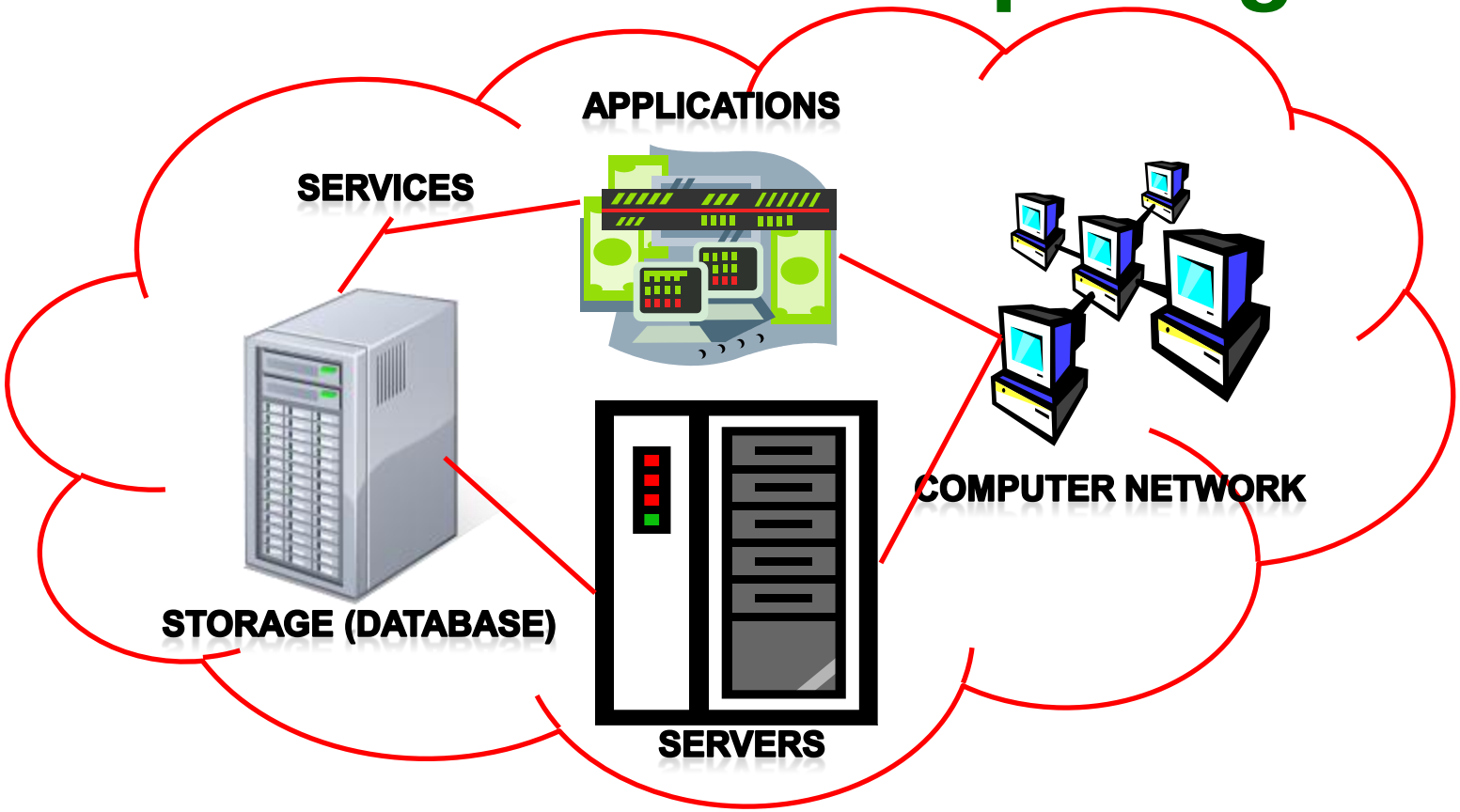
What is Cloud Computing?

What is Cloud Computing?

- “Cloud Computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources […] that can be rapidly provisioned and released with minimal management effort or service provider interaction.” (NIST)

Essential Characteristics according to NIST:

What is Cloud Computing



- Shared pool of configurable computing resources
- On-demand network access
- Provisioned by the Service Provider

Modes of Clouds

Public Cloud

- Computing infrastructure is hosted by cloud vendor at the vendors premises.
- and can be shared by various organizations.
- E.g. : Amazon, Google, Microsoft, Sales force

Private Cloud

- The computing infrastructure is dedicated to a particular organization and not shared with other organizations.
- more expensive and more secure when compare to public cloud.
- E.g. : HP data center, IBM, Sun, Oracle, 3tera

Hybrid Cloud

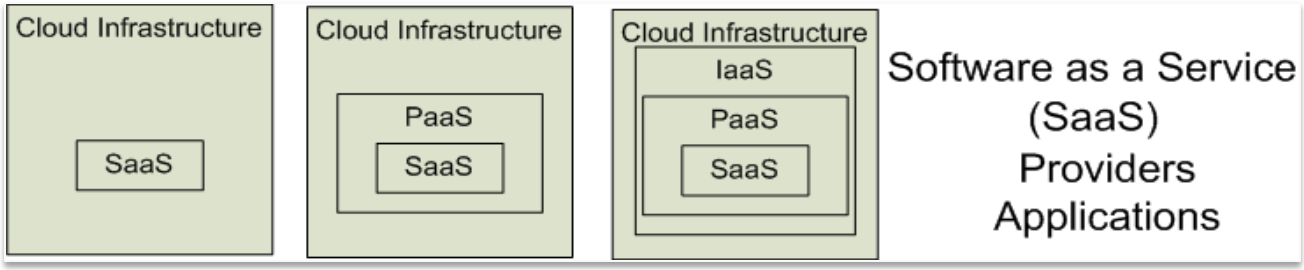
- Organizations may host critical applications on private clouds.
- where as relatively less security concerns on public cloud.
- usage of both public and private together is called hybrid cloud.

Cloud Service Models

Software as a Service (SaaS)	Platform as a Service (PaaS)	Infrastructure as a Service (IaaS)

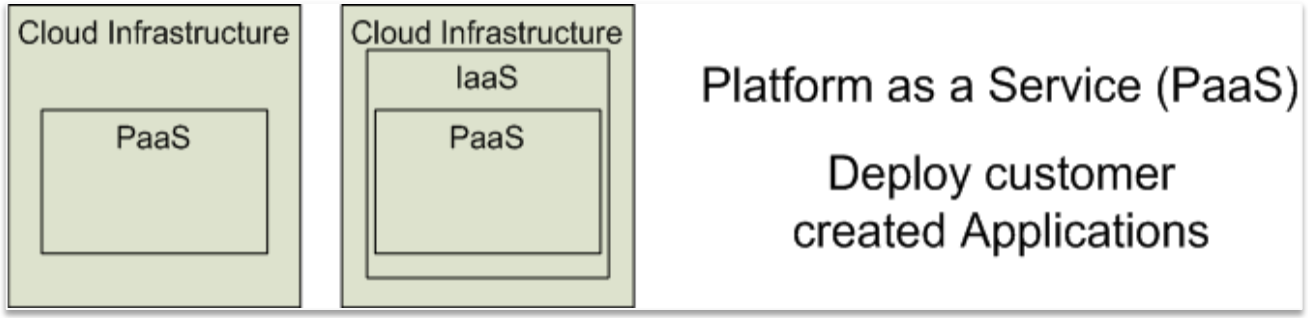
SalesForce CRM

LotusLive



Google App

Windows Azure
The Future Made Familiar



amazon web services™

rackspace HOSTING

computer NETWORKS

Example: Google App Engine

- Platform to develop and host web applications on Googles servers: PaaS
- Dynamic allocation: If requests for an app increase, the GAE allocates more resources to the app
- SLA: 99.5% availability
- Service for free, but resources are restricted:
  - Max 28 CPU hours per day
  - 1GB storage space in the High Replication Datastore
  - Mail API: Max 100 Emails per day
  - Other resources: need to buy
- Apps are sandboxed and run across multiple servers

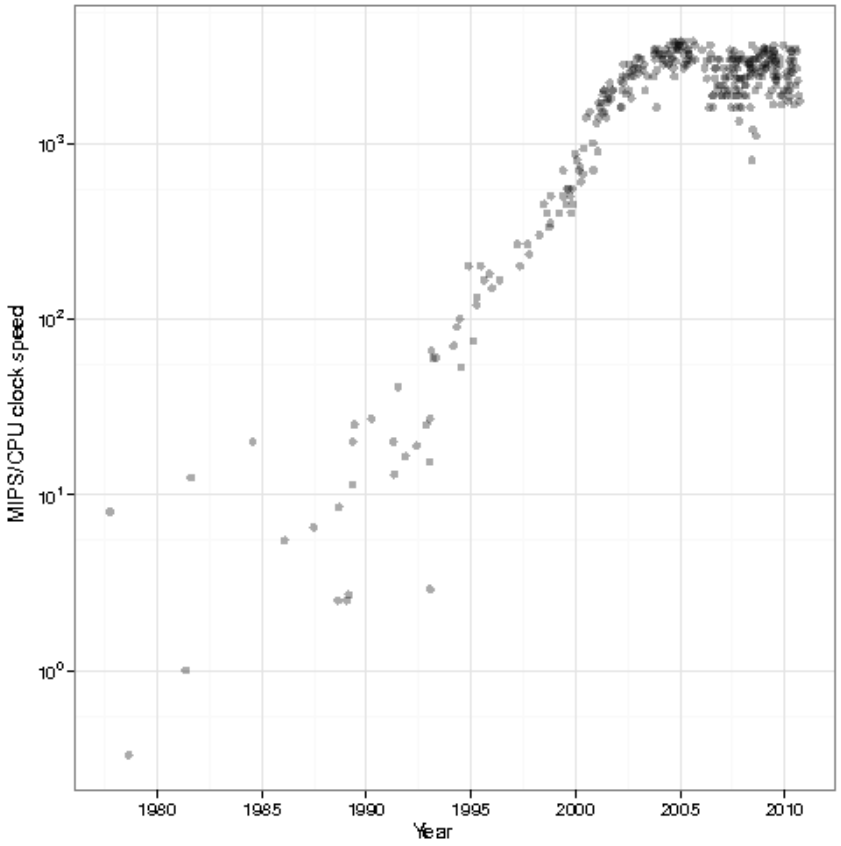
Example: Google App Engine

- Difference to, e.g., AmazonEC2 as IaaS:
  - Platform provides more infrastructure
  - Handles the deployment to cluster, monitoring, failover of app instances
  - Requires developers to stick with supported languages (currently Python and Java), APIs and Frameworks

Why do we need Cloud Computing?

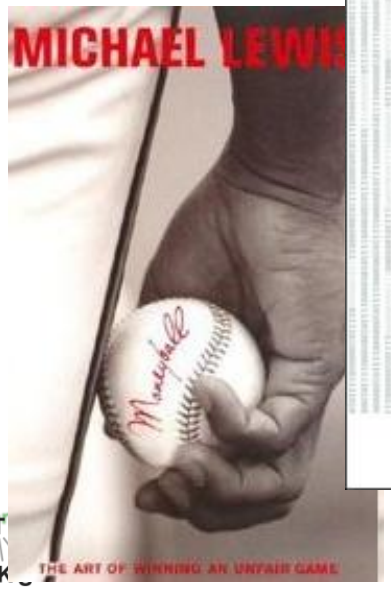
Background of Cloud Computing

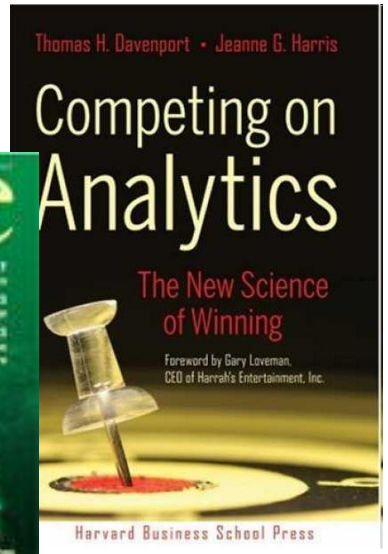
- 1990: Heyday of parallel computing, multi-processors
  - 52% growth in performance per year!
- 2002: The thermal wall
  - Speed (frequency) peaks, but transistors keep shrinking
- The Multicore revolution
  - 15-20 years later than predicted, we have hit the performance wall



At the same time…

- Amount of stored data is exploding…

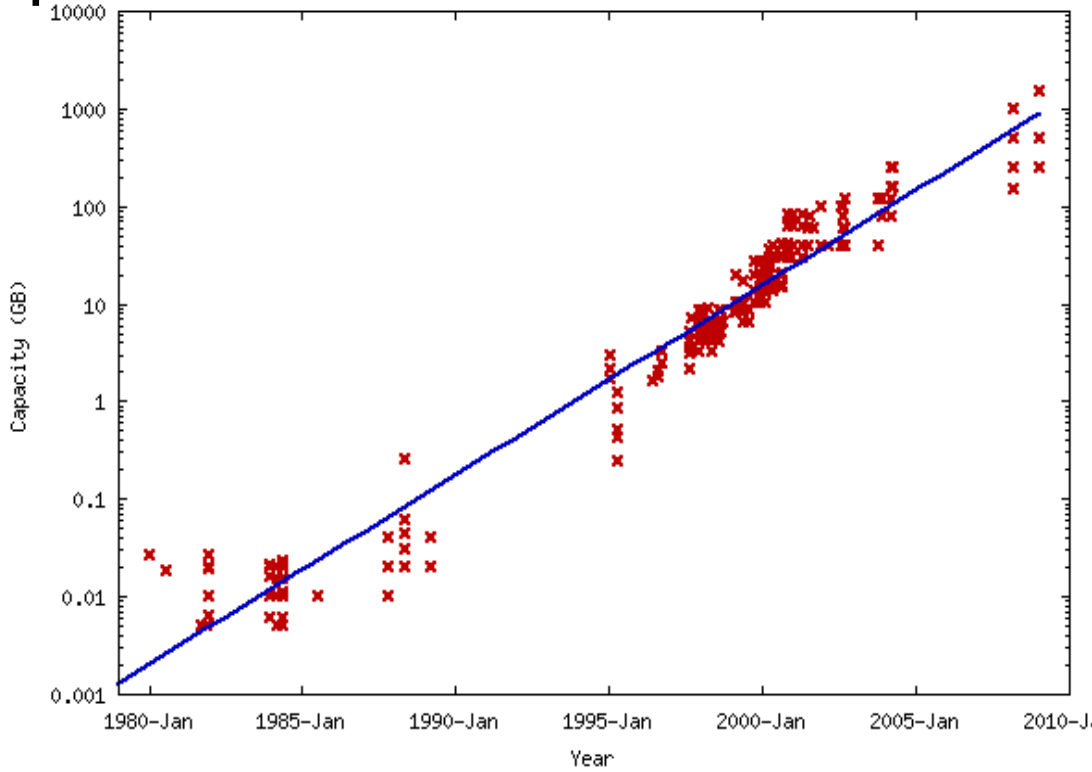




Data Deluge

- Billions of users connected through the net
  - WWW, FB, twitter, cell phones, …
  - 80% of the data on FB was produced last year

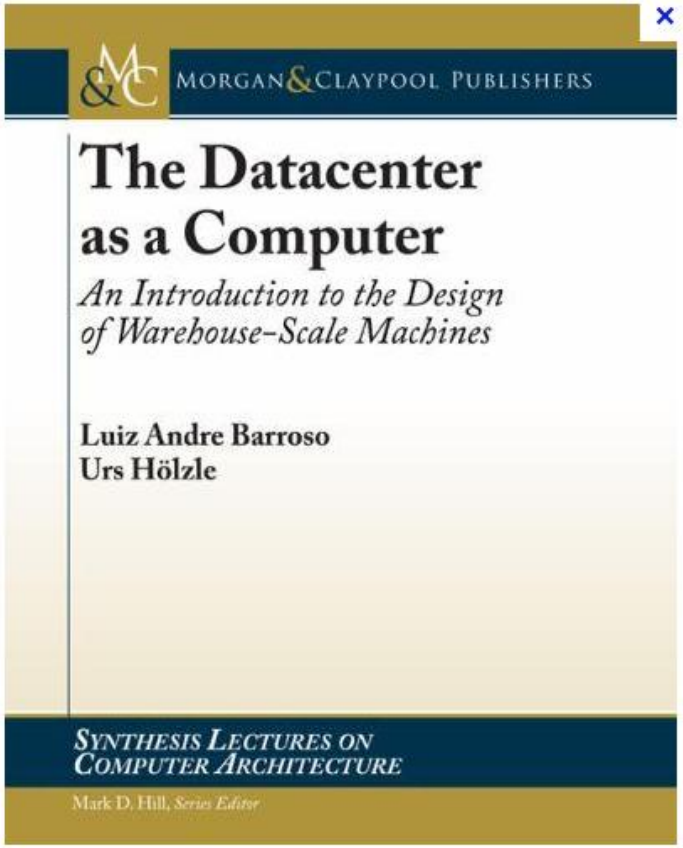
- Storage getting cheaper
  - Store more data!



Solving the Impedance Mismatch

- Computers not getting faster, and we are drowning in data
  - How to resolve the dilemma?
- Solution adopted by web-scale companies
  - Go massively *distributed* and *parallel*

The Datacenter is the new Computer



- *"Program"* == Web search, email, map/GIS, …
- *"Computer"* == 10,000's computers, storage, network
- Warehouse-sized facilities and workloads
- *Built from less reliable components than traditional datacenters*

Why Cloud Computing?

- Customer's perspective:

https://www.youtube.com/watch?v=yMJ75k9X5_8

- Also from a researchers perspective:
  - Possibility to run intense experiments
  - Offloading of parts of your work to the cloud, collect results
  - Our university can use GWDG data center for various purposes
    - GWDG cloud storage
    - GWDG computing clusters (for simulations etc.)

How is Cloud Computing implemented?

Characteristics

Common Characteristics:

Massive Scale	**Resilient Computing**
Homogeneity	**Geographic Distribution**
Virtualization	**Service Orientation**
Low Cost Software	**Advanced Security**

Virtualization, in computing, refers to the act of creating a virtual (rather than actual) version of something, including but not limited to a virtual computer hardware platform, operating system (OS), storage device, or computer network resources.

Virtualization

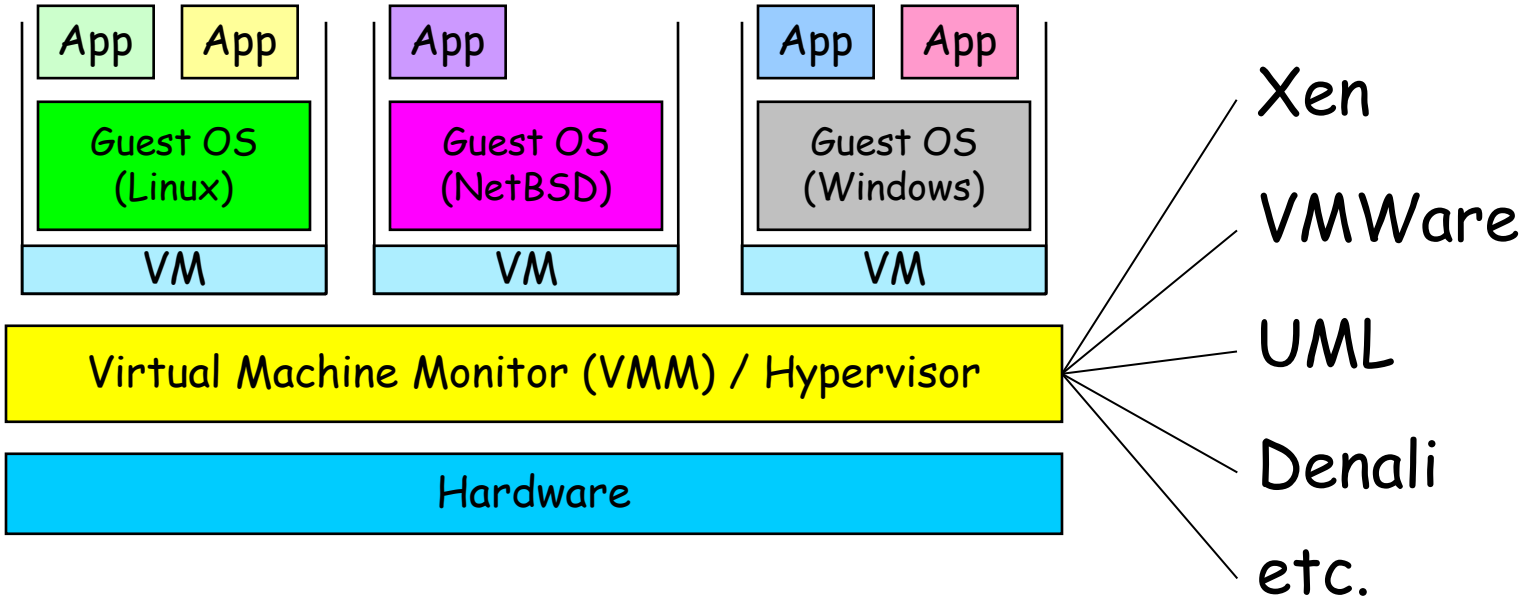
- Virtual workspaces:
  - An abstraction of an execution environment that can be made dynamically available to authorized clients by using well-defined protocols,
  - Resource quota (e.g. CPU, memory share),
  - Software configuration (e.g. OS, provided services).
- Implemented on Virtual Machines (VMs):
  - Abstraction of a physical host machine,
  - Hypervisor intercepts and emulates instructions from VMs, and allows management of VMs,
  - VMWare, Xen, etc.

Virtualization - Purposes

- *Abstraction* – to simplify the use of the underlying resource (e.g., by removing details of the resource's structure)
- *Replication* – to create multiple instances of the resource (e.g., to simplify management or allocation)
- *Isolation* – to separate the uses which clients make of the underlying resources (e.g., to improve security)

Virtual Machines

- VM technology allows multiple virtual machines to run on a single physical machine (also: *multi-tenancy*!).



Two Types of Hypervisors

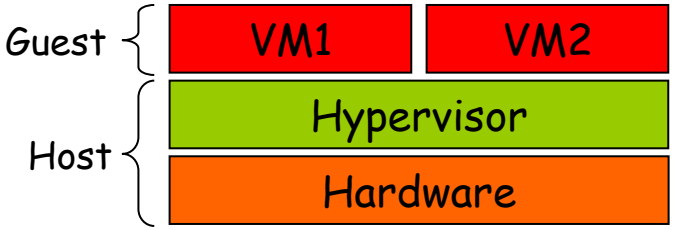
Native

(also called: bare-metal)

Has complete **control over hardware**

Doesn't have to **"fight" an OS**

Type 1 (native/bare-metal)



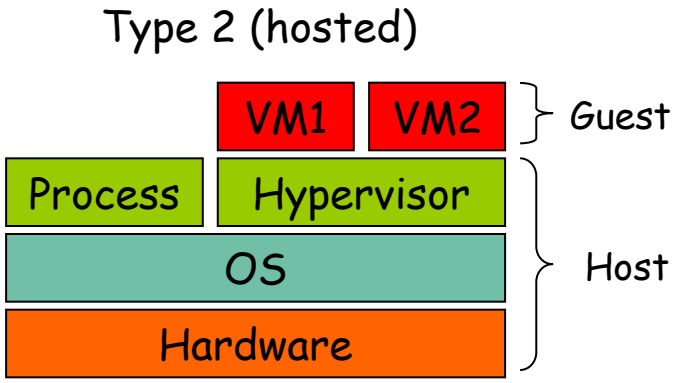
VMware ESX, Microsoft Hyper-V, Xen

Two Types of Hypervisors

Hosted

Avoid **code duplication**: need not code a **process scheduler**, **memory management** system – the **OS already does** that

Can run native **processes alongside** VMs

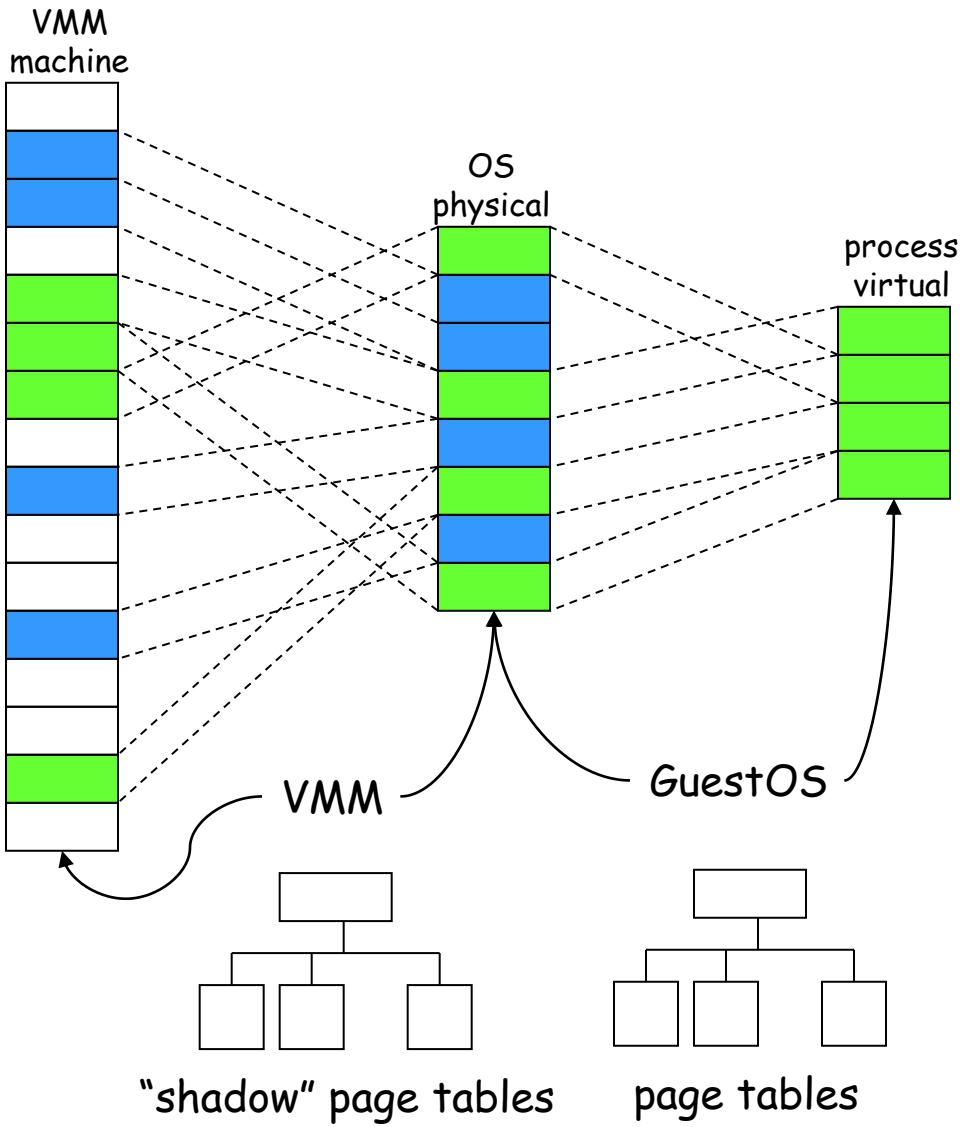


VMware Workstation, Microsoft Virtual PC, Sun VirtualBox, QEMU, KVM

CPU Virtualization Approaches

- Full Virtualization:
  - Complete decoupling of guest OS from underlying hardware
  - OS not aware it is virtualized, sensitive calls: binary translation
  - Simplest, most secure approach
- Para-Virtualization
  - Guest OS kernel is modified to enable communication with hypervisor (hypercalls)
  - Allows for improvement in performance and efficiency
  - But: low portability, high maintenance overhead
- Hardware-assisted Virtualization:
  - Privileged and sensitive calls are set to automatically trap the HV
  - No need for translation or para-virtualization

Memory Virtualization



- Isolation/protection of Guest OS address spaces

Load Balancing

- The cloud needs to match performance requirements of applications
- Load balancing: distribute incoming application requests across multiple resources
  - Can scale to very high demand by adding resources (performance)
  - Allows for continuation of service in case of resource failure (reliability)
- Uses standard balancing algorithms (as seen e.g. in CN)
  - Round robin, WFQ, priority, overflow
  - Also more network specific: low latency, least connections

Load Balancing: Persistance

- Load balancing can route successive requests from an application to different resources

- How does each resource know the state of the application? Persistance mechanism based on sessions!

Load Balancing: Persistance

- Three types:
  - **Stickiness**: all requests belonging to a user session routed to the same server – what if server fails (no failover!)?
  - **Session database**: store session in external, replicated session database – overhead? (example: ASP.net session DB)
  - **Client side**: store session info in browser cookies or URL rewrites – very efficient, but at times insecure (e.g., MITM attack)

Bottleneck: The Network

"The average cloud environment might have 50 dedicated servers to one admin, and what you really need to get to is 500 servers to one admin, or what happened in the case of Microsoft, 10,000 servers. Without automation we don't have speed and scale - the very reason we want to go to." (Microsoft)

- Complex with standard IP networking (mainly manual processes: have to manually configure each device with physical presence!)

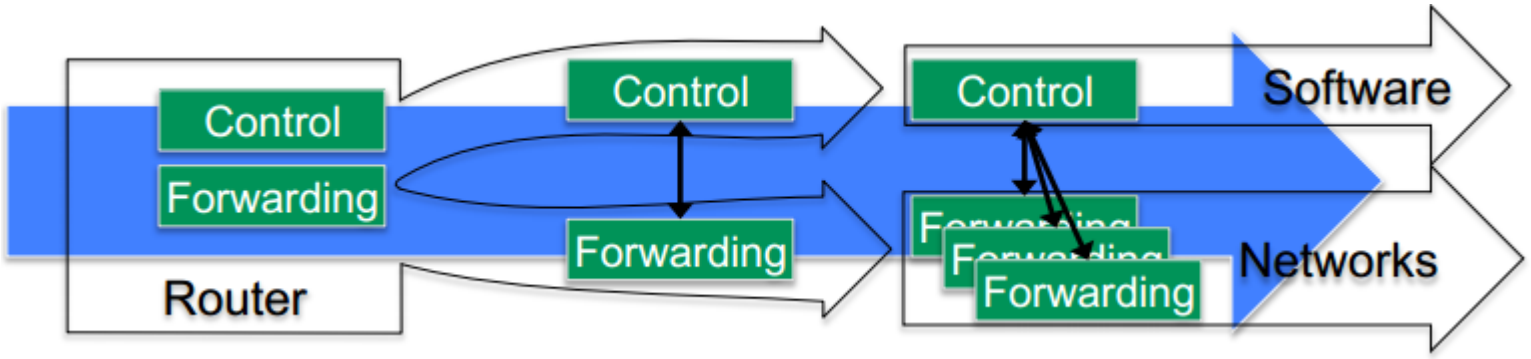
Bottleneck: The Network

"Even simple topologies take days or weeks to create. Workload placement and mobility are restricted by physical network limitations and hardware dependencies require vendor-specific expertise. Network configuration is performed manually and maintenance is both expensive and resource-intensive." (VMWare)

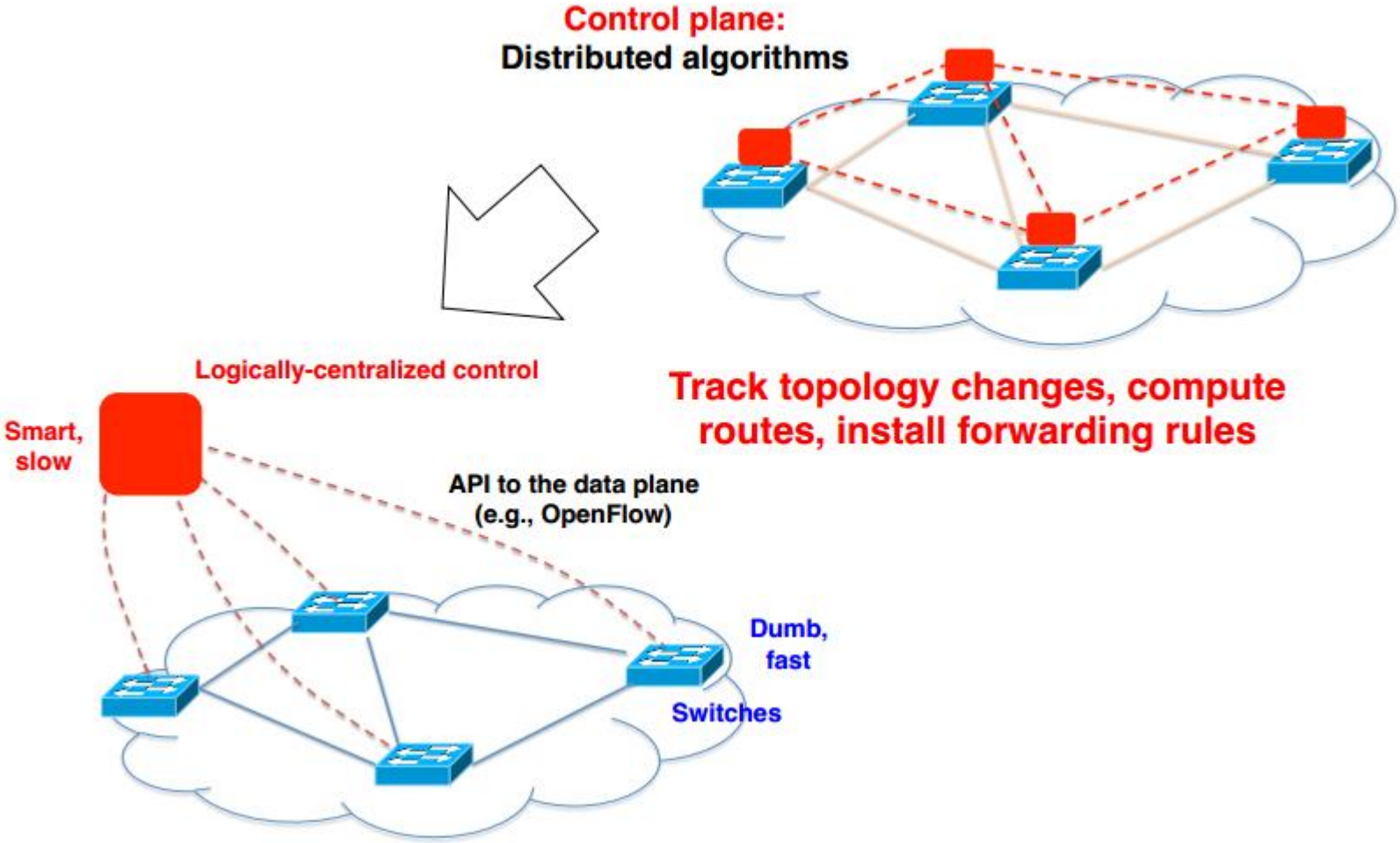
- We need automation!

Software Defined Networking

- Solution: Software-Defined-Networking (SDN)
  - Decouples the control plane from the data plane



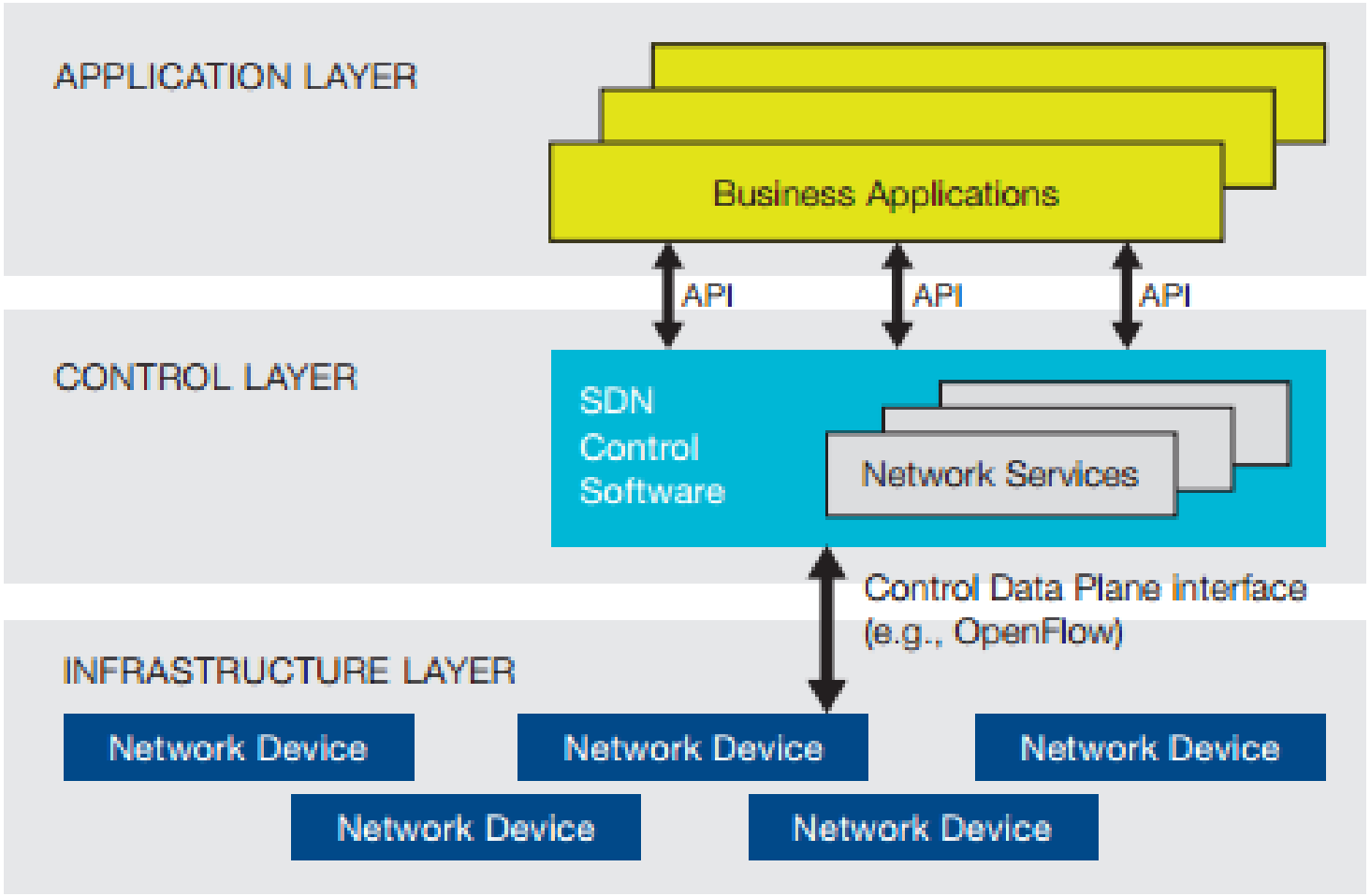
Software Defined Networking



Software Defined Networking

- SDN makes the network *programmable*
- OSPF, DiffServ, IntServ, MPLS, RSVP?
  - All such protocols can be done in software, controlled by a central instance
  - Scalable, easily managable, better interoperability

SDN Components at a glance



SDN Components at a glance

- Programmable Open APIs:
  - Connects applications with control plane
  - Allows for *programming* of routing, QoS, etc.

- Standard Communication Interface (e.g., OpenFlow):
  - Between control and data planes
  - Allows direct access to forwarding plane

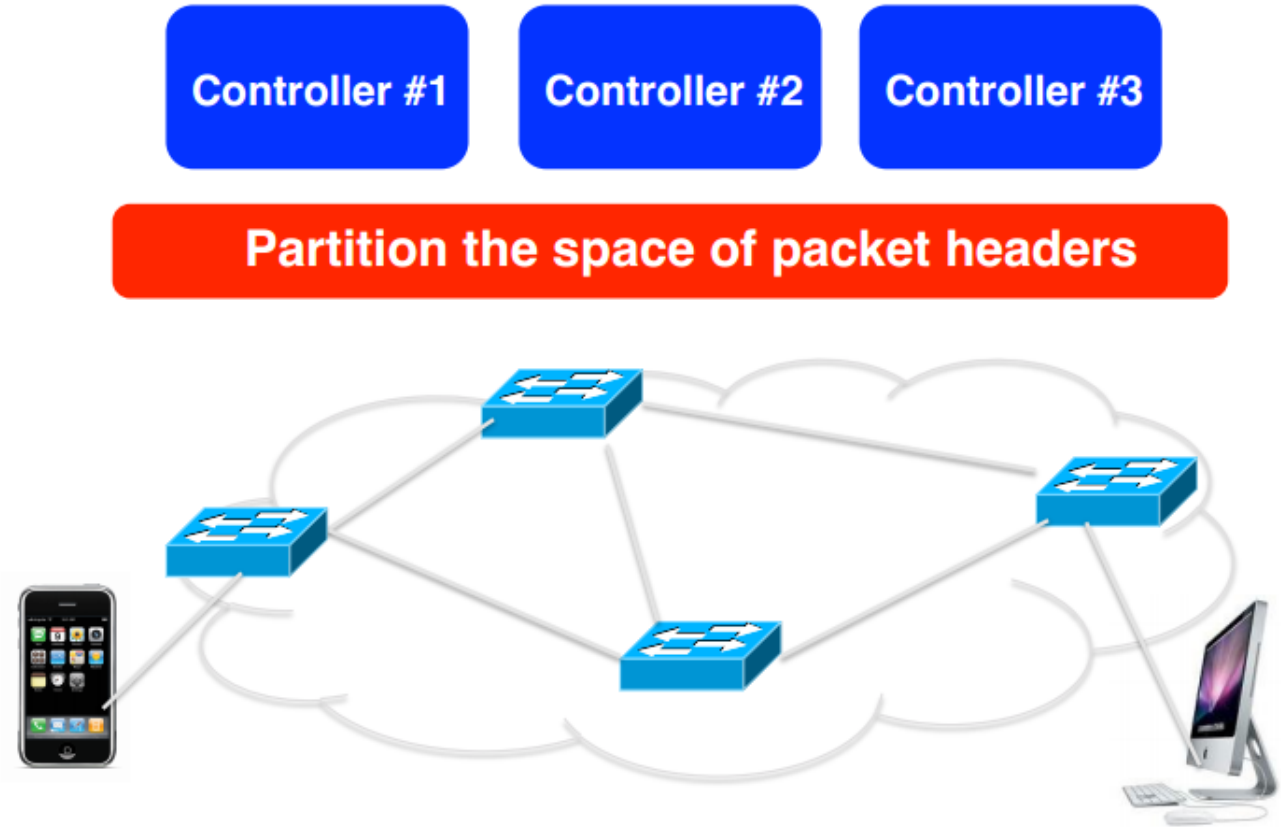
Materials by the Open Networking Foundation:
https://www.opennetworking.org/

SDN Components at a glance

- Network Controller (*logically* centralized):
  - Positioned between both above, has full control over the data plane
  - Sets up rules, actions, etc. for the network devices
  - Core element of SDN

SDN and Cloud?

- E.g., network virtualization

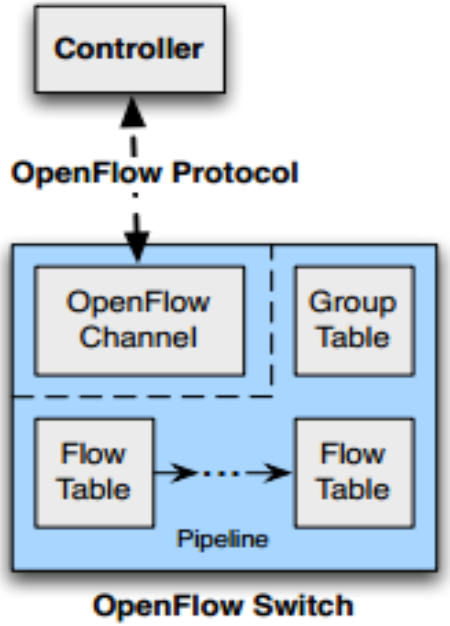


SDN and Cloud!

- In the cloud, SDN further allows for…
  - elastic resource allocation (e.g., to match QoS agreements)
  - distribution of the load on links (e.g., between backbone and application servers in SaaS)
  - scalability (no need to manually configure each of thousands (or even millions?) of devices)
  - overhead reduction
  - …and more

OpenFlow – The SDN Protocol

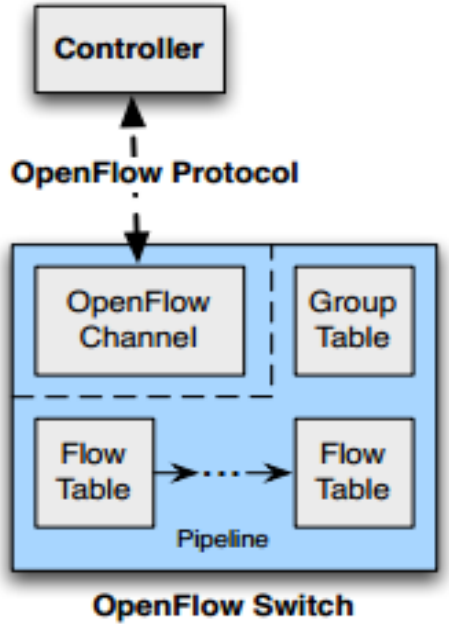
- Communication between the controller and the network devices (i.e., switches)



Specification by the Open Networking Foundation:
https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.4.pdf (March 2014)

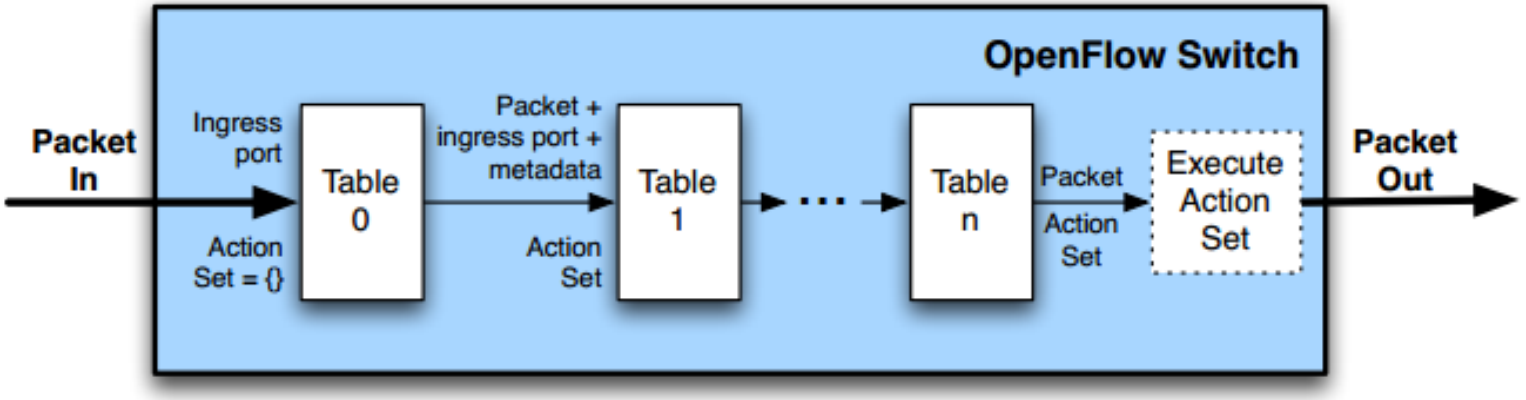
OpenFlow – The SDN Protocol

- Main components: *Flow* and *Group Tables*
  - Controller can manipulate these tables via the OpenFlow protocol (*add, update, delete*)
  - Flow Table: reactively or proactively defines how incoming packets are forwarded
  - Group Table: additional processing



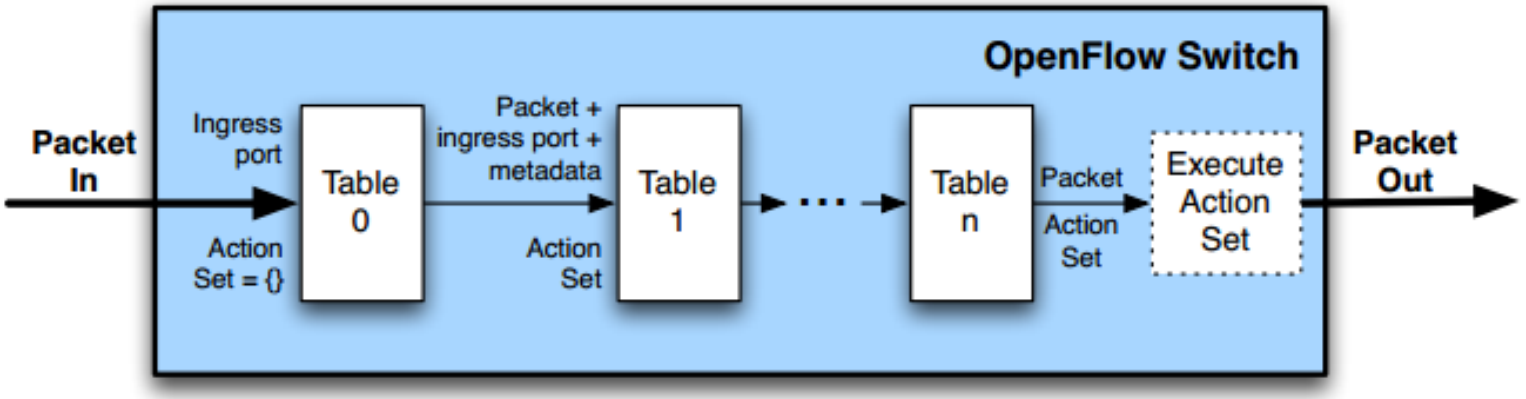
OpenFlow – Switches

- Two different versions of an OpenFlow Switch
  - *OF-only* (packets can only be processed by OF tables) and *OF-hybrid* (allow optional normal Ethernet handling (see CN lecture))
- OF-only: all packets go through a *pipeline*
  - Each pipeline contains one or multiple flow tables with each containing one or multiple *flow entries*

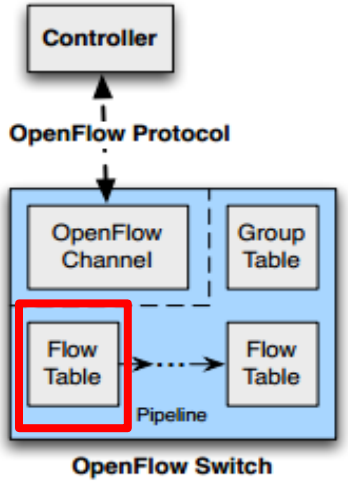


OpenFlow – Switches

- Incoming packets are matched against Table 0 first
- Find highest priority match and execute instructions (might be a Goto-Table instruction)
- Goto: Only possible forward



OpenFlow – Switches



- Flow Table entry structure:

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags

- Match fields: where matching applies (i.e., ingress port, packet headers, etc.)
- Priority: matching precedence of flow entry
- Counters: update on packet match with entry
- Instructions: what to do with the packet
- Timeout: max idle time of flow before ending

OpenFlow – Switches

- Flow Table entry structure:

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags

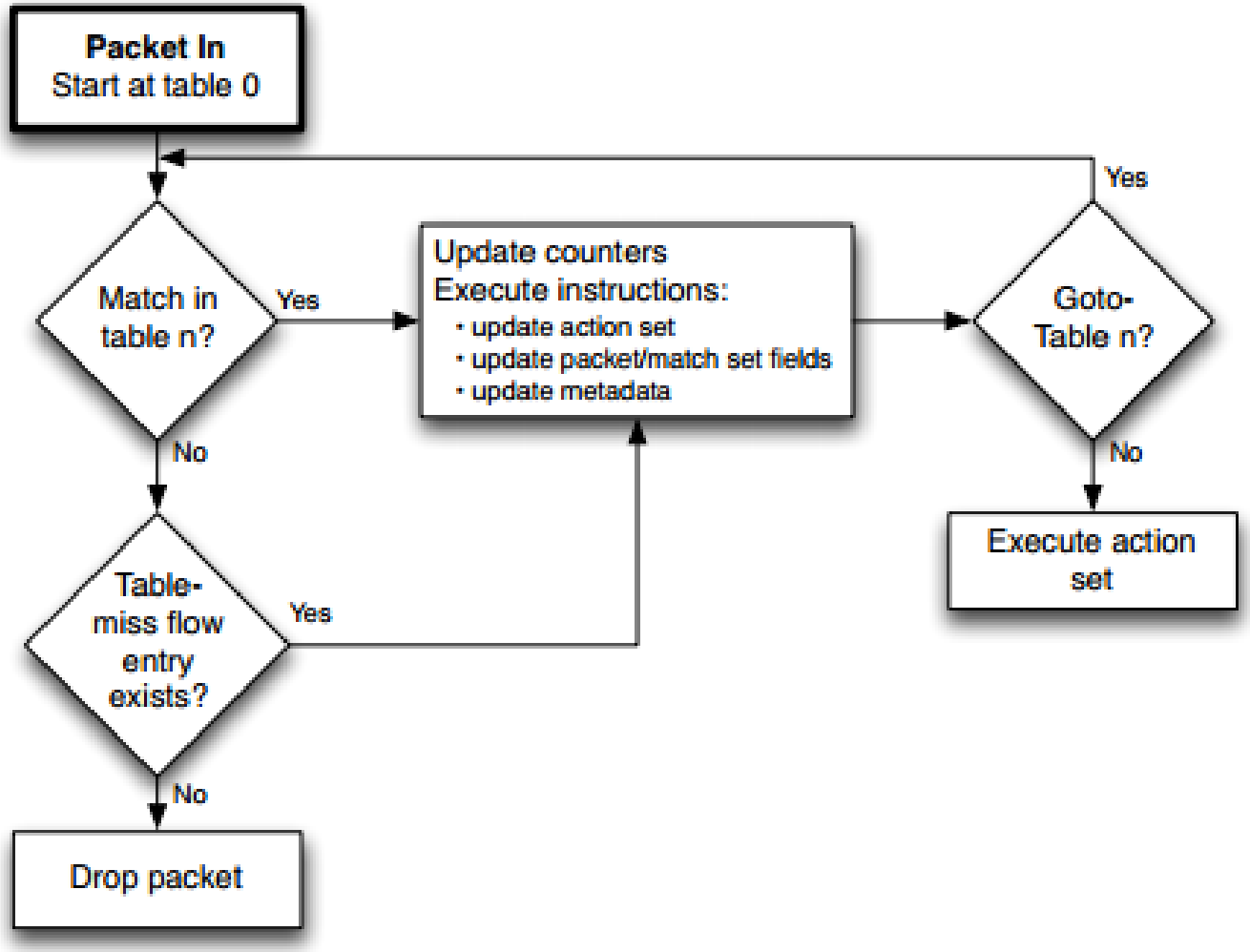
- Match fields: where matching applies (i.e., ingress port, packet (IP, eth) headers, etc.)
- Priority: matching precedence of flow entry

- A flow entry with all match fields as wildcard and priority 0: *table miss* entry

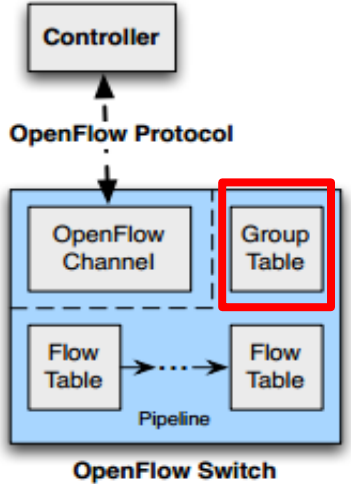
OpenFlow – Switches

- If no match in table: *table miss*
- Handling: depends on table configuration – might be *drop packet, forward to other table, forward to controller*
- Forward to controller allows to set up a flow entry (i.e., at the beginning of a flow)

OpenFlow - Matching



OpenFlow – Switches



- Group Table entry structure:

Group Identifier	Group Type	Counters	Action Buckets

- Group Identifier: 32-bit ID to uniquely define group on the switch (locally)
- Group Type: *indirect/all/fast failover/select*
  - Specifies which *action bucket* is executed
- Counters: update on packet processed
- Action Buckets: ordered list of buckets, each containing a *set* of instructions

OpenFlow – Switches

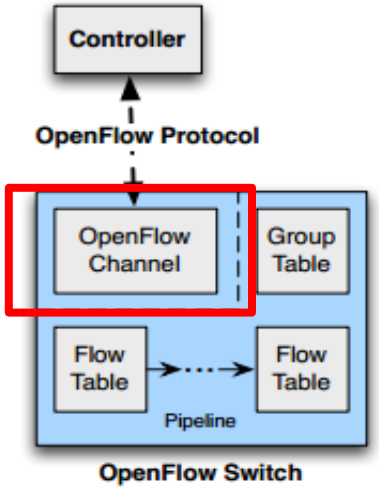
- Group Table entry structure:

Group Identifier	Group Type	Counters	Action Buckets

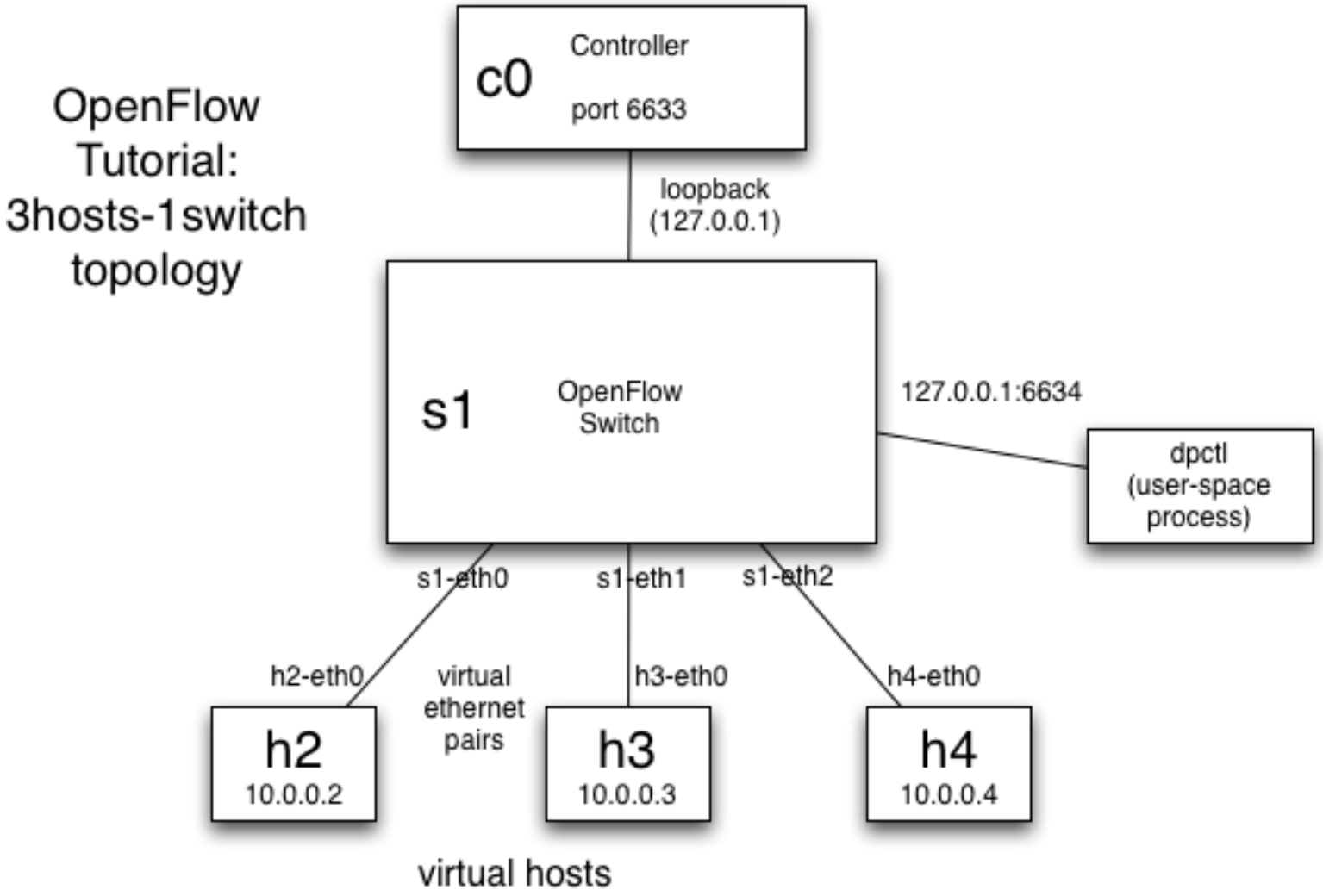
- Group Tables allow for more complex forwarding
  - E.g., multicast: use *all* group type to execute all action buckets (packet will be cloned for each bucket, and then forwarded through the instruction set)

OpenFlow – OpenFlow Channel

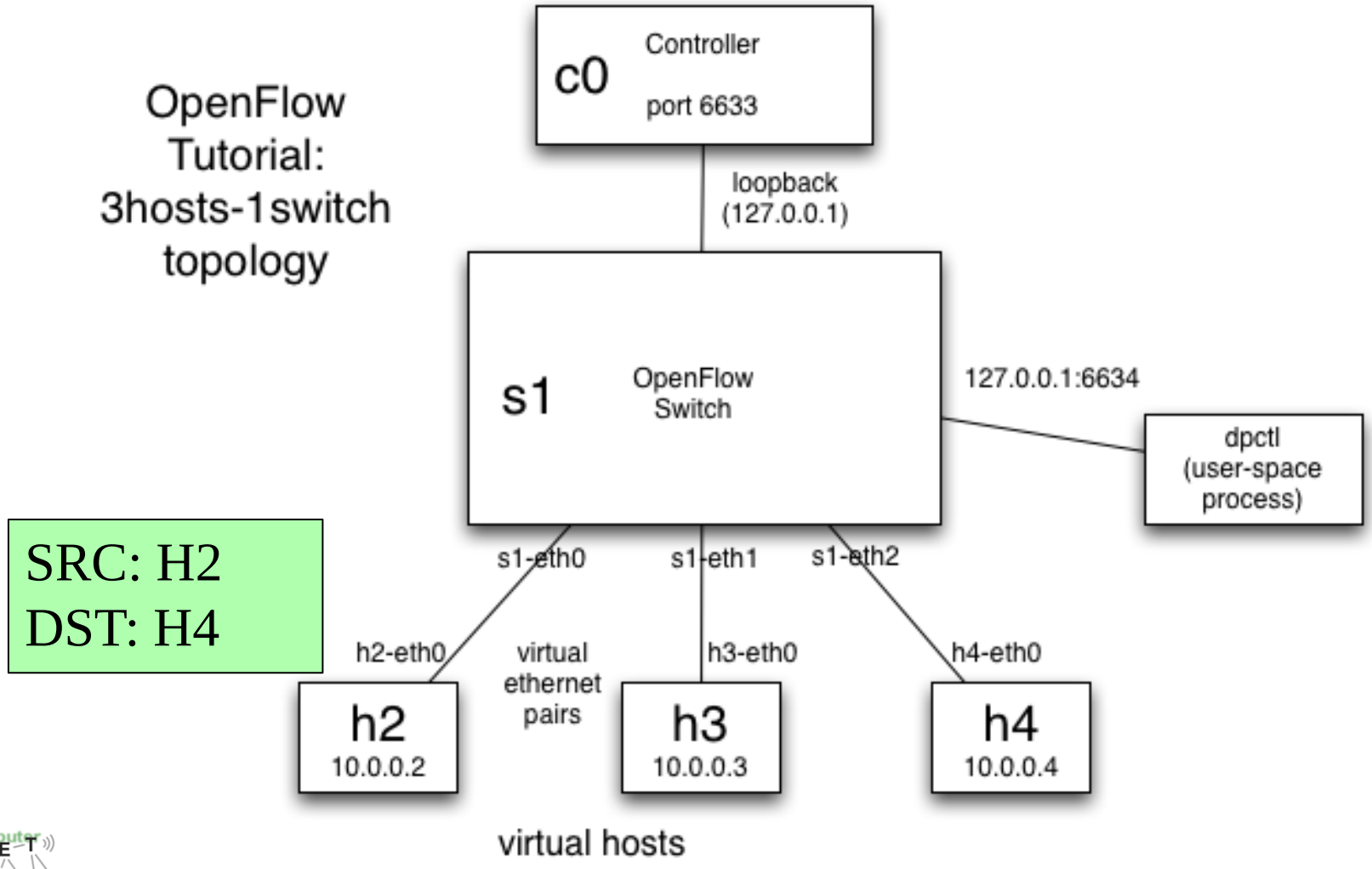
- Different message types available:
  - *Controller-to-Switch*, *Asynchronous* or *Symmetric*
- Controller-to-Switch:
  - Lets the controller control the switch
  - E.g., *Modify-State* command to manipulate flow tables
- Asynchronous:
  - Switch-to-controller requests (e.g., at table miss)
- Symmetric:
  - May be sent from both ends (e.g., echo command)



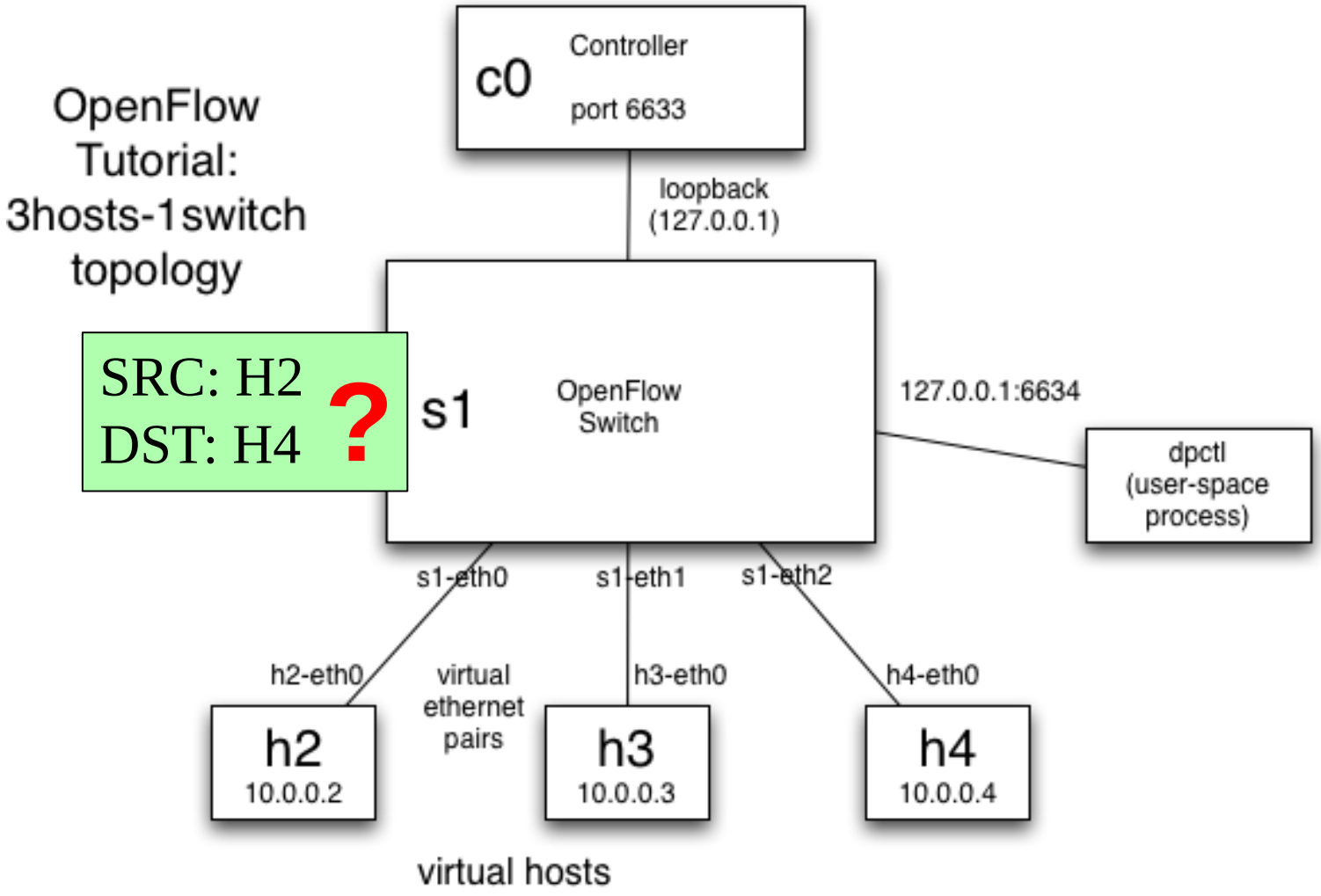
OpenFlow - Example



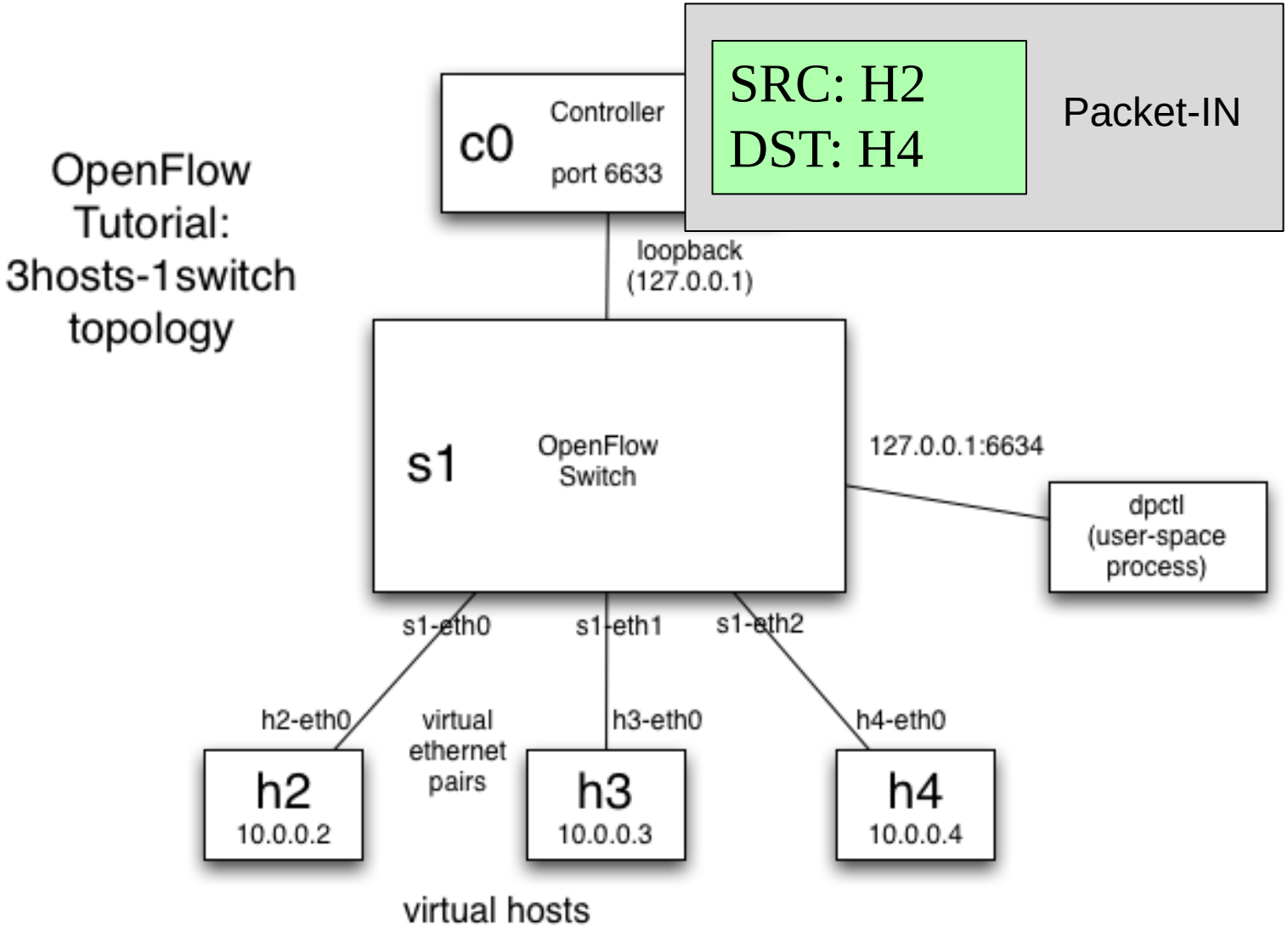
OpenFlow - Example



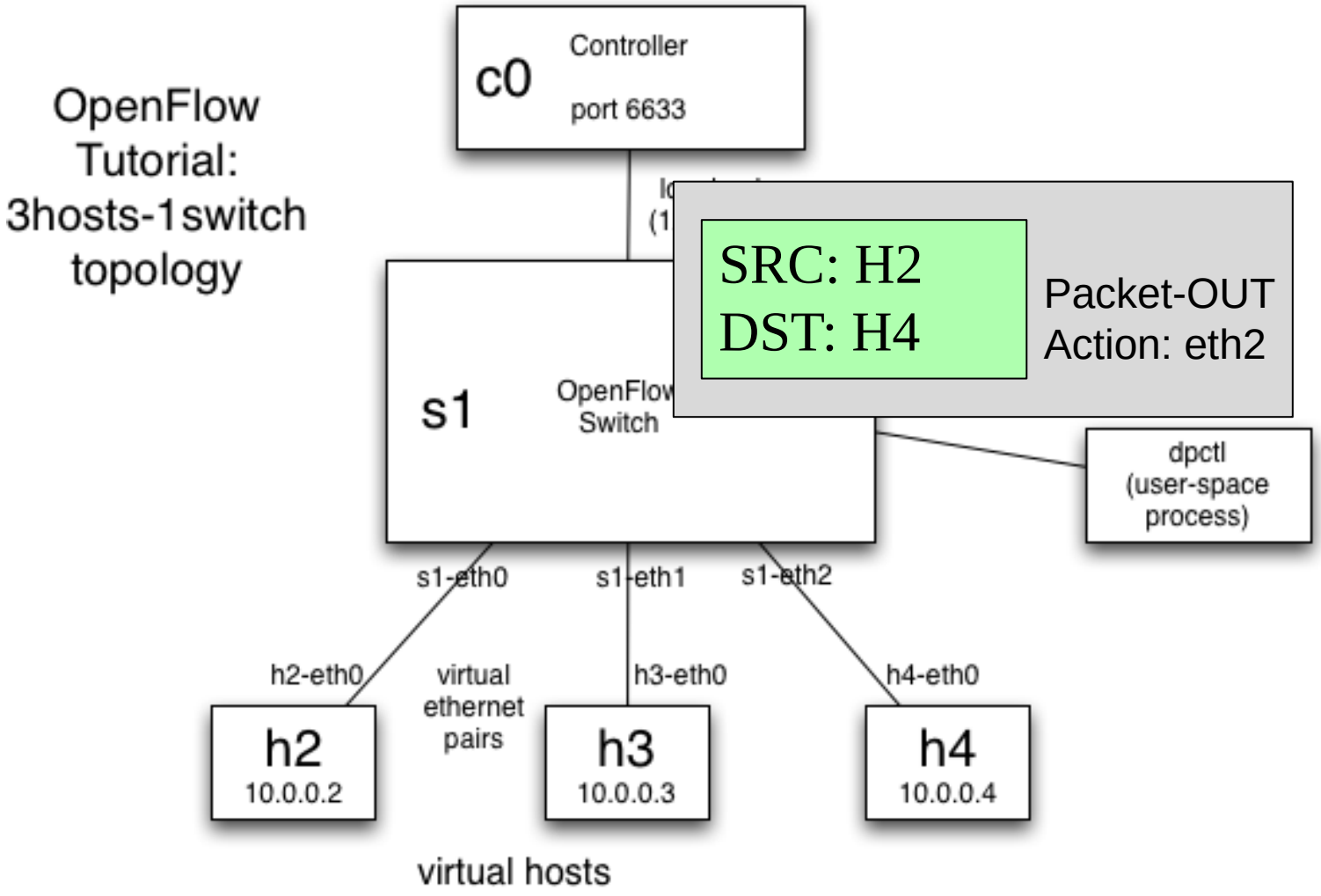
OpenFlow - Example



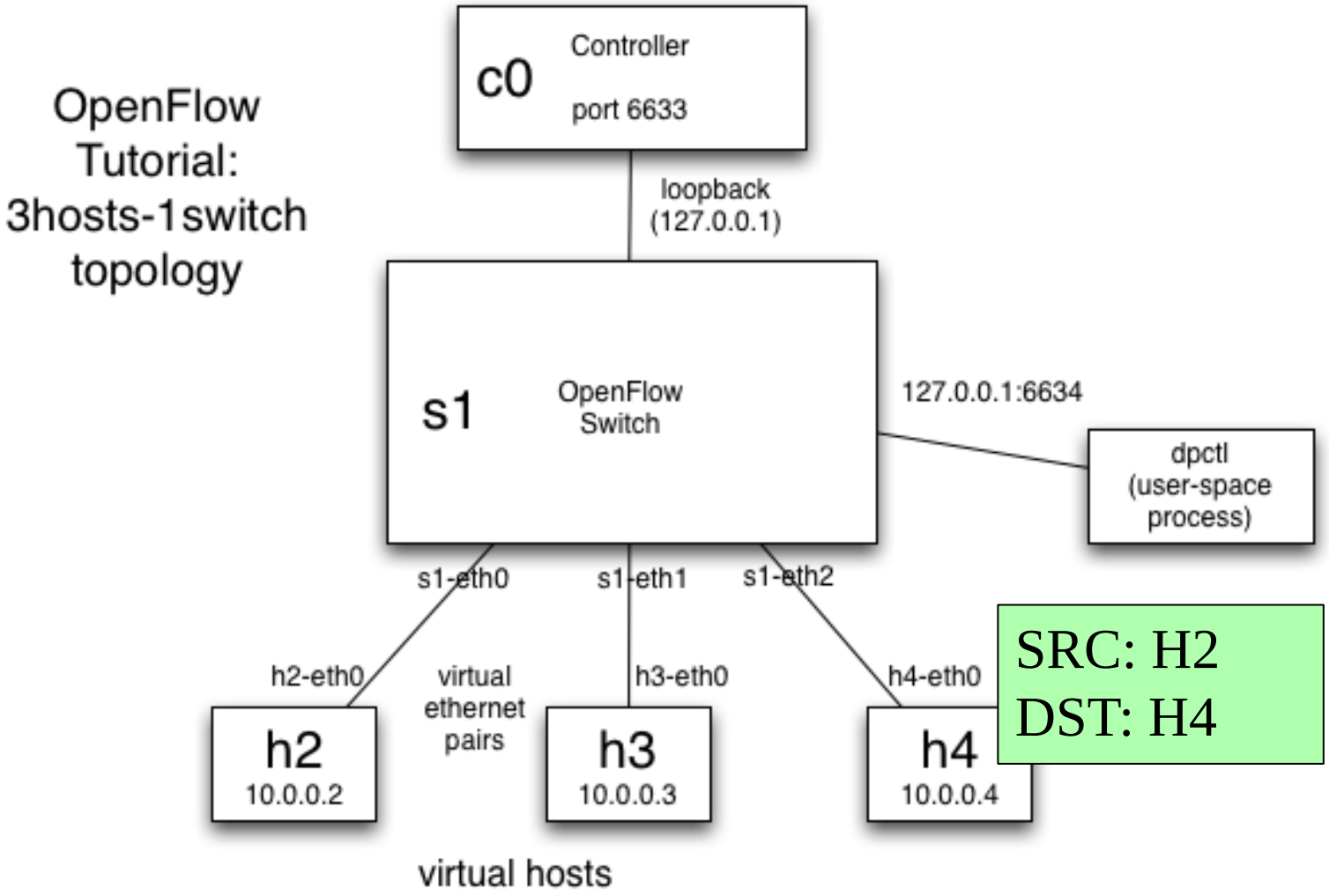
OpenFlow - Example



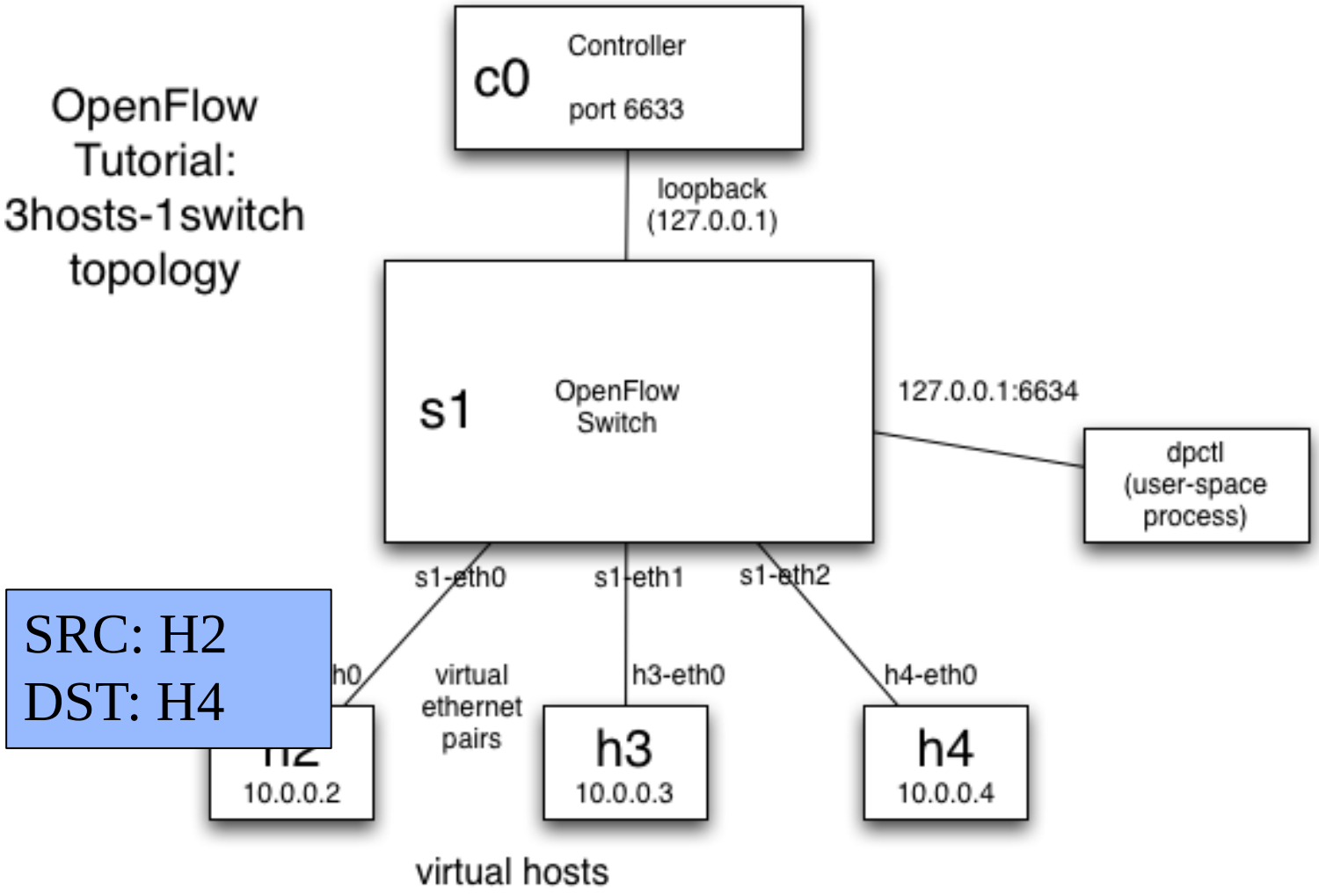
OpenFlow - Example



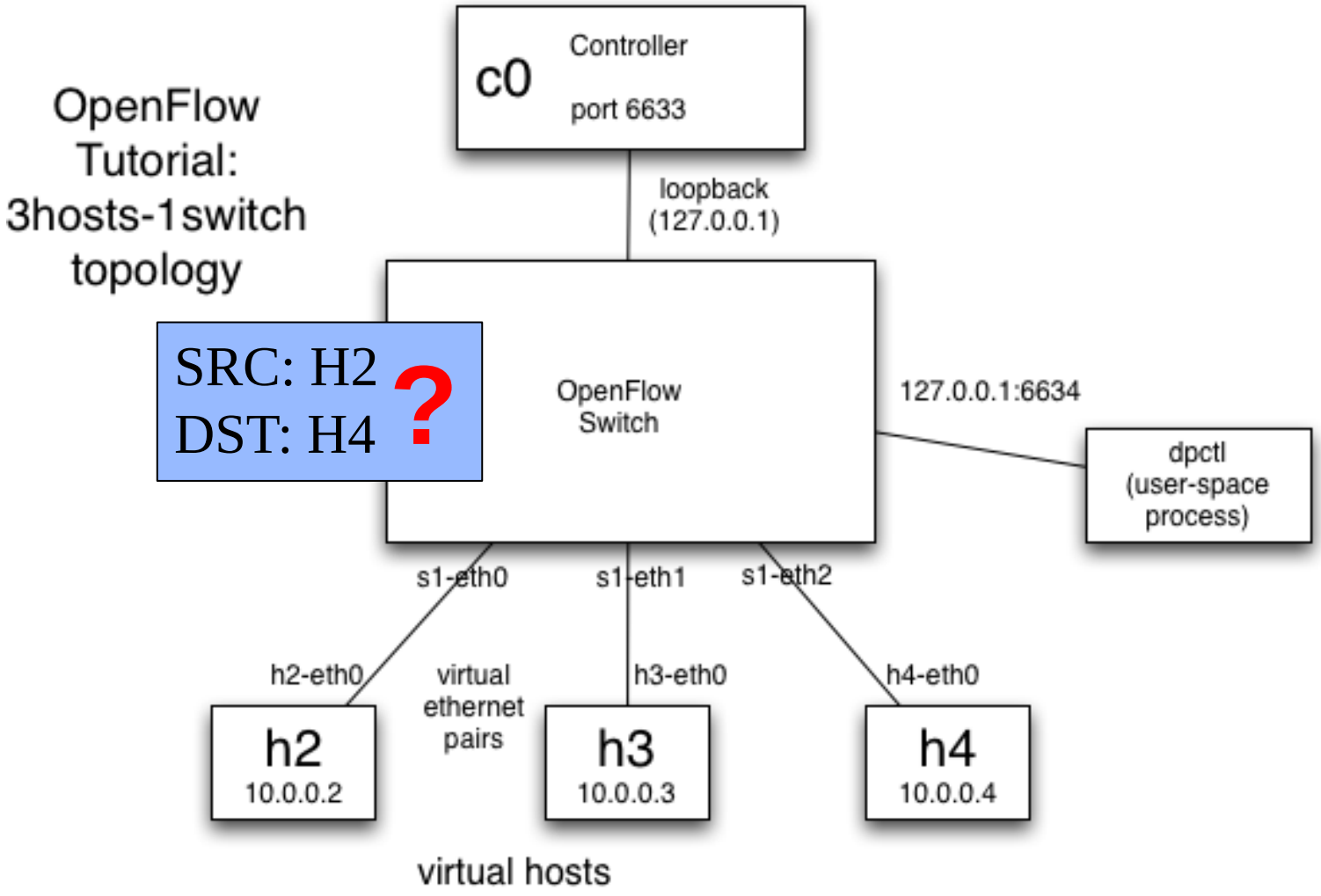
OpenFlow - Example



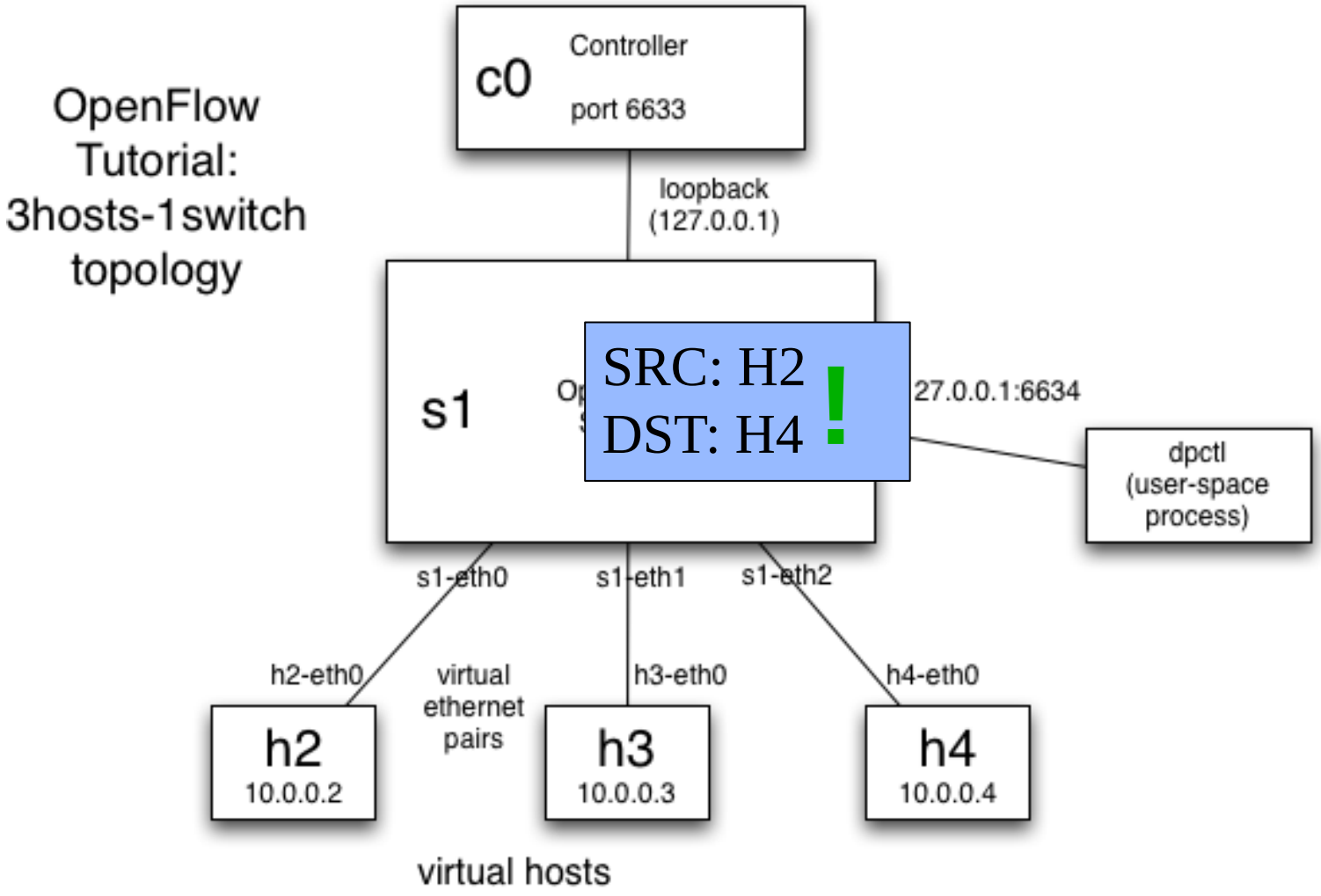
OpenFlow - Example



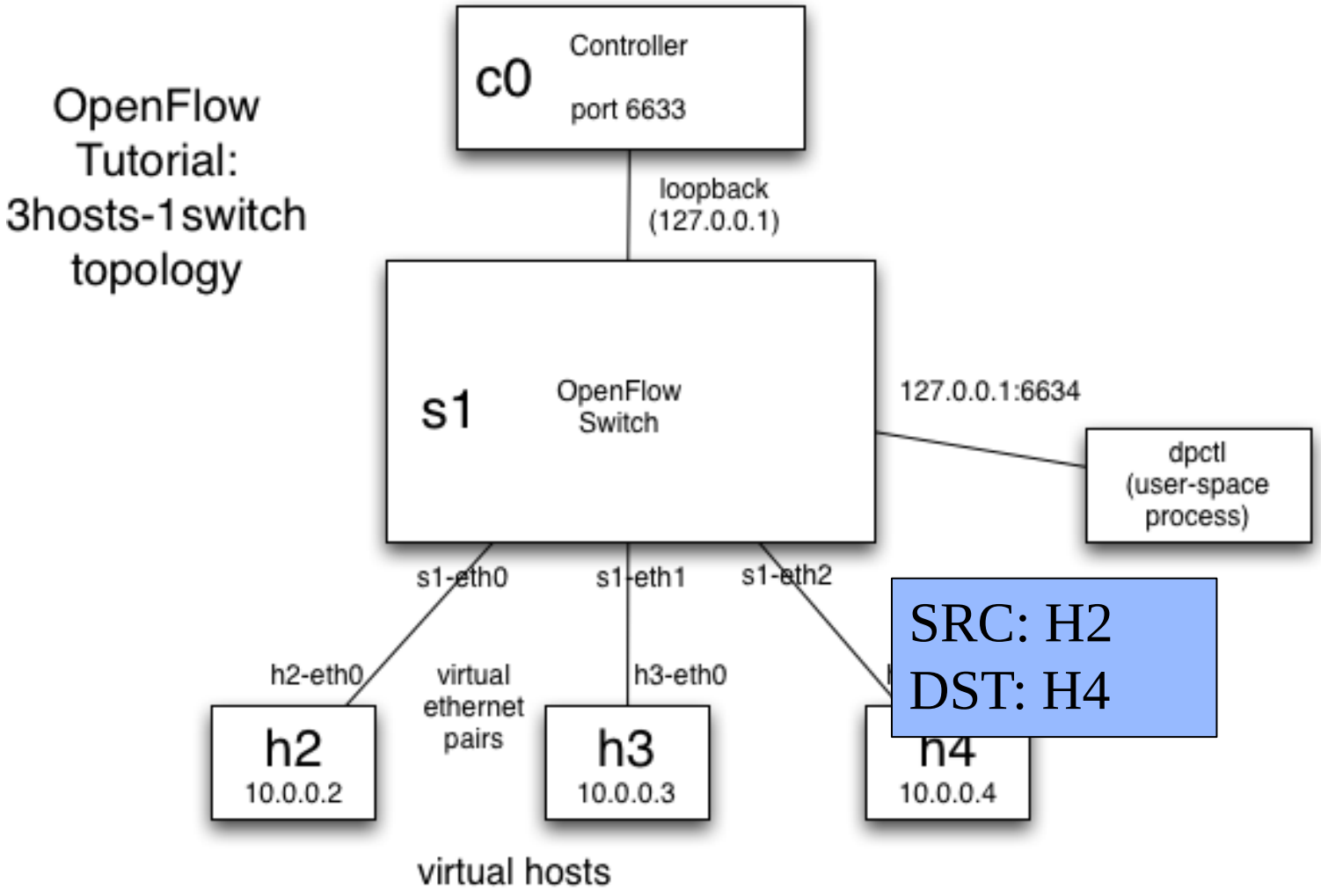
OpenFlow - Example



OpenFlow - Example



OpenFlow - Example



If you're interested…

- https://www.opennetworking.org/
- "Why SDN is going to change everything" (Bethany Mayer, SVP of Hewlett Packard (HP))
  http://h17007.www1.hp.com/de/de/networking/solutions/technology/sdn/#sdn-video
- N. McKeown, INFOCOM 2009 keynote on SDN:
  http://www.cs.rutgers.edu/~badri/552dir/papers/intro/nick09.pdf

What's in the Cloud Ecosystem?

- Data sharing
  - File systems like Google File System (GFS), Hadoop File System (HDFS), …
- Data analysis & programming abstractions
  - Google MapReduce, PIG, Hive, Spark, …
- Multiplexing of resources & coordination
  - Mesos, YARN (MRv2), ZooKeeper, BookKeeper, …
- (DataBases
  - Cassandra (No-SQL without single point of failure))